DOCUMENT RESUME

ED 111 441 88 IR 002 553

AUTHOR Laudato, Nicholas C.; Roman, Richard A.
TITLE Computer-Assisted Instruction in Word Problems: Design and Assessment.
REPORT NO PU-LRDC-1975/12
PUB DATE Apr 75
NOTE 38p.; Paper presented at the American Educational Research Association Annual Meeting (Washington, D. C., March 30-April 3, 1975)

EDRS PRICE MF-$0.76 HC-$1.95 Plus Postage
DESCRIPTORS *Arithmetic; *Computer Assisted Instruction; Educational Research; Elementary Education; Information Processing; *Instructional Design; Mathematics Instruction; Models; Problem Solving
IDENTIFIERS Word Problems

ABSTRACT

A computer-based curricular package was developed to teach elementary school students to solve arithmetic word problems and to teach problem solving skills applicable to situations not involving word problems. An information processing model for solving word problems was used to sequence the problems. Pilot test results with five students suggested that the sequence derived from the model was hierarchical, that students learned from the program, and that they enjoyed the experience. Statistical data includes test material, tryout data, and a list of 22 references. (Author)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ED111441

SCOPE OF INTEREST NOTICE

The ERIC Facility has assigned this document for processing to: IR SE

In our judgement, this document is also of interest to the clearinghouses noted to the right. Indexing should reflect their special points of view. CS

U S DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

LEARNING RESEARCH AND DEVELOPMENT CENTER

COMPUTER-ASSISTED INSTRUCTION IN WORD PROBLEMS: DESIGN AND ASSESSMENT

1975/12

NICHOLAS C. LAUDATO AND RICHARD A. ROMAN

IR 002 553

University of Pittsburgh

COMPUTER-ASSISTED INSTRUCTION IN WORD PROBLEMS: DESIGN AND ASSESSMENT

Nicholas C. Laudato and Richard A. Roman

Learning Research and Development Center

University of Pittsburgh

Paper presented at the meeting of the American Educational Research Association, Washington, D. C., April 1975.

The research reported herein was supported by a grant from the National Science Foundation (NSF-GJ-540X) and by the Learning Research and Development Center, supported in part as a research and development center by funds from the National Institute of Education (NIE), United States Department of Health, Education, and Welfare. The opinions expressed in this paper do not necessarily reflect the position or policy of the sponsoring agencies, and no official endorsement should be inferred.

Abstract

This paper describes the development and pilot test results of a computer-based curricular package dealing with instruction in word problem solving for elementary school students. An information processing model for solving word problems is used to sequence the problems. Pilot test results with five students suggest that the sequence derived from the model is hierarchical, that students learn from the program, and that they enjoy the experience.

COMPUTER-ASSISTED INSTRUCTION IN WORD PROBLEMS: DESIGN AND ASSESSMENT

Nicholas C. Laudato and Richard A. Roman

University of Pittsburgh

This paper describes the development and pilot test of a curricular package dealing with instruction in word problem solving for the elementary school. The design and rationale for the curricular package are detailed in another paper (Roman & Laudato, 1974). The curriculum, designated the Word Problem Program, is one of several programs developed to investigate uses of the computer for instruction in problem solving. It was created as part of the Computer-Assisted Instruction in Problem Solving Project, with the fundamental goal of teaching elementary school children some generalizable problem solving skills. This project is, in turn, conducted as part of the Oakleaf Computer Resource Project initiated by the Learning Research and Development Center in 1969 under sponsorship of the National Science Foundation. The principal purpose of the Oakleaf Computer Resource Project is to "investigate appropriate and effective uses of the computer in an individualized school in order to foster the adoption of individualized systems of elementary education" (Block, Carlson, Fitzhugh, Hsu, Jacobson, Puente, Roman, Rosner, Simon, Glaser, & Cooley, 1973, p. 1).

The Word Problem Program is designed to accomplish two instructional goals: to teach elementary school children how to solve arithmetic word problems, and to teach problem solving skills applicable to situations not involving word problems. All instruction in the Word Problem Program is accomplished by means of a computer program.

All the topics presented in this paper are discussed in more detail in a doctoral dissertation by Laudato (1975). Because the scope of this study was so vast, the writers were forced to limit the number of topics and their depth of discussion in its presentation. The topics presented were selected in an attempt to both summarize the study and highlight its significance. This paper describes the goals of the program, a methodology for defining and sequencing instructional objectives based on an information processing model, and the design features that enable the program to meet these goals. Finally, it summarizes the results of an experimental pilot test.

Goals of the Word Problem Program

A word problem is a written representation of a concrete, physical situation that requires the problem solver to engage in purely mental activities in order to reach a quantitative solution. In the Word Problem Program, the problem solving task is viewed as requiring the application of two distinct sets of skills for solution: word problem-specific skills and general problem solving skills. To specify the word problem-specific skills, we have designed a model of human behavior on word problem solving tasks based on the information processing work of Bobrow (1968) and Paige and Simon (1966). According to the model, people solve word problems by:

(a) translating individual phrases into symbolic expressions.

(b) combining the expressions to form equations.

(c) manipulating (if necessary) the equations to attain a directly solvable equation.

(d) performing arithmetic computations to attain the answer.

While the language used to describe the model is algebraic, elementary school students do not use formal algebra to solve word problems.

The algebraic language, then, is a metaphor for the processes of symbolic representation and manipulation that actually do take place.

The information processing model clarifies the skills required to solve word problems. The skills of the third and fourth steps are usually algorithmic to elementary school students, while the skills of the first two steps include higher level cognitive processing. In the Word Problem Program, concern is with problem solving and not with competence in algorithmic manipulation, and thus students are assumed capable of solving number sentences and computing accurately. The first major goal of the Word Problem Program is to provide an environment that facilitates the acquisition and application of skills necessary to perform the first and second steps of the model.

Several researchers (Newell & Simon, 1972; Olton, 1969; Papert, 1972; Polya, 1957; Wickelgren, 1974) have hypothesized skills and strategies for solving word problems whose application is not limited to the specific case of arithmetic word problems, but are generalizable to many situations. These general problem solving skills include recognizing the existence of a problem solving situation and clearly stating a goal; gathering, recording, and organizing the data; breaking a complex problem into parts, solving the parts, and reintegrating the partial solutions into a solution to the whole problem; and recognizing when the solution has been attained, when help is needed, or when to give up. The second major goal of the Word Problem Program is to provide an environment that facilitates the acquisition and application of these skills.

Instructional Objectives

An instructional objective for the Word Problem Program consists of a group of problems that require the same kinds of information processing skills. Thus, in order to specify an instructional objective, we need to specify the skills that the student must be able to apply and the criteria

for evaluating that application. Instructional objectives are sequenced so that later objectives require quantitatively more and/or qualitatively different processing skills. The specification of skills for each objective and, consequently, the hierarchical ordering of objectives, is facilitated by a component analysis performed with respect to the word problem-specific skills identified by the model. To accomplish the component analysis, we have devised an algorithm which solves the problems used in the Word Problem Program and outputs data about the processing steps required to reach a solution. These processes are analogous to the skills humans employ to solve word problems.

The solution algorithm is a set of rules that translates a given word problem into symbolic form and manipulates the symbols to attain a solution to the problem. The algorithm operates on item forms. An item form specifies a set of problems which have identical syntactic form (Hively, 1963; Osburn, 1968). In this case, each item form specifies a set of problems created by filling several grammatical slots with items from word lists. For example, given the following word list and sample item form, 108 (4 x 3 x 3 x 3) unique word problems can be generated without varying the numerical data.

Word Lists:

(NAME)	=	JOHN, SUE, JOAN, DICK
(VERB)	=	COLLECTED, OBTAINED, GATHERED
(OBJECT)	=	STAMPS, APPLES, BALLS
(DATE)	=	AUGUST, THE SPRING, MAY
(PRONOUN)	=	HE, SHE

Sample Item Form:

DURING (DATE), (NAME) (VERB) (NUMBER, 1) (OBJECT).
SINCE THEN, (PRONOUN) (VERB) (NUMBER, 2) MORE (OBJECT).
HOW MANY (OBJECT) DOES (PRONOUN) HAVE NOW?

Example Problems:

DURING THE SPRING, JOHN COLLECTED 25 STAMPS.
SINCE THEN, HE COLLECTED 13 MORE STAMPS.
HOW MANY STAMPS DOES HE HAVE NOW?

DURING MAY, SUE OBTAINED 16 APPLES.
SINCE THEN, SHE OBTAINED 32 MORE APPLES.
HOW MANY APPLES DOES SHE HAVE NOW?

In the Word Problem Program, there are 45 different word lists, each containing from 3 to 10 words. Thus, the number of unique word problems that can be generated from each item form reaches the hundred thousands. (A typical item form that uses six lists with approximately seven words per list can generate $7^6 = 117,649$ different problems.) In addition, the specific numbers used in each problem vary to further diversify the problems. A more extended discussion of the use of item forms is found in Roman and Laudato (1974).

A set of approximately 200 item forms defines the universe of problems for use in the Word Problem Program. These problems require from one to three different arithmetic operations applied from one to five times. They are roughly equivalent in difficulty to those from the Individualized Mathematics and Individually Prescribed Instruction materials for grades 3 to 7 (Research for Better Schools & Learning Research and Development Center, 1972).

The solution algorithm is applied in two stages: a translation stage and a manipulation stage. The translation stage makes a symbolic representation of each phrase in the item form and passes the resulting equations to the manipulation stage. The manipulation stage combines the equations in such a way as to produce a solution. The following sections illustrate the operation of each of these stages. For simplicity, the examples will show the algorithm operating on problems instead of item forms.

The Translation Stage

The first task that the solution algorithm must perform in the translation stage is that of dividing the problem into phrases. Under specified conditions, certain words and/or punctuation serve as phrase delimiters, e.g., periods, question marks, or the word "IF" followed by text that is, in turn, followed by "AND" or "HOW MANY." After phrases have been identified, the algorithm proceeds to translate each phrase into a symbolic representation. Each phrase is interpreted as either assigning a numerical value to a variable or asking a question. The algorithm ignores phrases that do neither of these. Thus, a phrase such as "JIM AND TOM WALKED TO THE ZOO" would not be represented symbolically in the resultant set of equations.

Some word problems require temporal information in order to interpret and solve them. The algorithm notes the occurrence of phrases such as "TWO WEEKS AGO," "YESTERDAY," "NOW," and so on, and tags each expression as to the relative temporal sequence of the events in the problem. If there are no such temporal references in the problem, this information is omitted.

Most phrases assign a piece of numerical data to some object, person(s), or event in the problem situation. For example, the phrase

(1) JIM BOUGHT 15 APPLES

assigns to a variable named "the number of apples of Jim" the value "15." This assignment is represented symbolically as

(2) (APPLES, JIM) = 15

Similarly, the phrase

(3) TOM BOUGHT 5 MORE APPLES THAN JIM

assigns to a variable named "the number of more apples than Jim of Tom"

the value "5." This is represented as

(4) (MORE APPLES THAN JIM, TOM) = 5

Both of these translations follow the general rule which states that if the phrase consists of a string of words (represented as <STRING.1>) followed by a verb which connotes positive acquisition (represented as <+VERB>) followed by a number (represented as <NUMBER.1>) followed by a second string of words (<STRING.2>), or,

(5) <STRING.1> <+VERB> <NUMBER.1> <STRING.2>

then such an expression is symbolically represented as

(6) (<STRING.2>, <STRING.1>) = <NUMBER.1>

Thus, if the algorithm finds a phrase which matches the form of (5), it translates it to (6). A phrase such as

(7) HOW MANY APPLES DID JIM AND TOM BUY?

must be viewed as

(8) JIM AND TOM BOUGHT ? APPLES

to be translated, by the same general rule, to

(9) (APPLES, JIM AND TOM) = ?

Some symbolic expressions in the form of (6) must be further translated. For example, expression (4) above,

(MORE APPLES THAN JIM, TOM) = 5

is transformed by a specific rule to

(10) (APPLES, TOM) = (APPLES, JIM) + 5

That is, the statement "the number of more apples than Jim of Tom is 5" is equivalent to the statement "the number of apples of Tom is equal to the number of apples of Jim plus 5."

Finally, the algorithm scans the set of generated expressions for further modification by means of specific rules such as the union rule. This rule states that

(11) (X, Y, AND Z) = (X, Y) + (X, Z), where X, Y and Z are text.

Application of this rule to expression (9) generates the additional expression

(12) (APPLES, JIM AND TOM) = (APPLES, JIM) + (APPLES, TOM).

The solution algorithm for the translation stage consists of a set of rules which perform such transformations on each phrase of every item form in the domain of the program. This stage is complete when each resultant expression has been reduced to its simplest form. The symbolic expressions are then passed to the manipulation stage.

The Manipulation Stage

The manipulation stage of the algorithm consists of a set of rules governing the combination and manipulation of the expressions derived by the translation stage. The first step in the manipulation stage is to locate the expression which contains the unknown. If this expression is in indirect form, it must be manipulated so that the question mark will be isolated on one side of the equation. An analogy to number sentences helps illustrate the concept of direct versus indirect equations. The number sentence "? = 25 + 19" is in direct form; no processing other than arithmetic is necessary in order to solve for "?". The number sentence "? - 5 = 12", however, is in indirect form; in order to attain the solution, it must first be transformed to the equivalent number sentence "? = 12 + 5" and then the value of "?" can be determined by arithmetic.

Several rules govern the manipulation of equations in order to attain desirable results. For example,

(13) (X, Y) + A = (X, Z) if and only if (X, Y) = (X, Z) - A

(14) (X, Y) · A = (X, Z) if and only if (X, Y) = (X, Z)/A,

where X, Y, Z are text and "A" is either a number or "?".

After the unknown has been isolated, the algorithm checks whether the equations are tagged according to temporal sequence. If this is so, an external equation is added to the previously generated list. This equation simply states that an event occurring in the past is modified by an event occurring over time to produce the current event, or

(15) (X, Y) [PAST] + (X, Y) [OVERTIME] = (X, Y) [PRESENT]

The algorithm now proceeds to make direct substitutions of expressions for those occurring in the equation that contains the unknown. If it locates, in the unknown equation, an expression for which no assignment is directly available, it searches for the presence of that expression in other equations. The algorithm must then proceed to manipulate the new equation to isolate the desired term. If this process does not succeed, other external equations may be added to the list of equations. For example, a multiplication equation which is often employed is

(16) (X, Y) · (Y, Z) = (X, Z)

This can be interpreted as follows: The phrases

(17) JOHN HAD 20 BOXES OF ORANGES.

(18) EACH BOX CONTAINED 10 ORANGES.

(19) HOW MANY ORANGES DID JOHN HAVE?

are translated to

(20) (BOXES, JOHN) = 20

(21) (ORANGES, BOX) = 10

(22) (ORANGES, JOHN) = ?

Applying equation (16) to equations (20), (21), and (22) yields

(23) (ORANGES, BOX) · (BOXES, JOHN) = (ORANGES, JOHN)

Two direct substitutions can now be performed on equation (23) to yield

(24) 10 • 20 = (ORANGES, JOHN) = ?

In applying the algorithm, each step is recorded to yield a "trace" of the process. Let us take as an example the phrases translated in the preceding subsection. The phrases were:

(1) JIM BOUGHT 15 APPLES.

(3) TOM BOUGHT 5 MORE APPLES THAN JIM.

(7) HOW MANY APPLES DID JIM AND TOM BUY?

These are easily translated to

(2) (APPLES, JIM) = 15

(4) (MORE APPLES THAN JIM, TOM) = 5

(9) (APPLES, JIM AND TOM) = ?

Before leaving the translation stage, equation (4) is further processed to yield

(10) (APPLES, TOM) = (APPLES, JIM) + 5

and equation (9) generates an additional equation by application of the union rule (11).

(12) (APPLES, JIM AND TOM) = (APPLES, JIM) + (APPLES, TOM)

In the manipulation stage, equation (9) is identified as the unknown. Since it is direct and does not involve sequence, a substitution is made of equation (12) into (9) to yield

(25) ? = (APPLES, JIM) + (APPLES, TOM)

Next, two direct substitutions of (2) and (10) into (25) yield

(26) ? = 15 + (APPLES, JIM) + 5

and finally, (2) is again substituted into (26) to give

(27) ? = 15 + 15 + 5 = 35

This problem thus required, in addition to the translation skills, the application of the union rule, the special "more than" rule, and four direct substitutions in order to attain the solution. The algorithm has been applied to each of the 200 item forms to produce lists of processing steps similar to that in the example above. The remainder of this section shows how these data were used to define and structure learning objectives.

The Objective Array

The solution algorithm outputs lists of the steps needed to solve a given item form. According to the model of human problem solving behavior, these steps are analogous to the information processing steps that humans utilize in solving word problems. In this sense, the application of the algorithm to each item form is equivalent to performing a component analysis. What was needed was a way to use information from the solution algorithm to create and sequence instructional objectives.

The list of steps from the solution algorithm was examined in detail to identify similarities and differences among item forms. It was immediately ascertained that some item forms were nearly identical in terms of the processing steps needed for solution. Such item forms should not differ significantly from each other in terms of either difficulty or prerequisite skills. It was thus decided that the item forms should be divided into a small number of groups that were, in theory, homogeneous with regard to information processing skills.

Several important distinctions between item forms were evident. First, the item forms differed on the basis of the number of direct substitutions required to solve them. This number varied from one to five. Item forms that require more direct substitutions require a greater amount

of processing and thus, hypothetically, are more difficult. Next, approximately half of the item forms required the use of rules to manipulate equations from indirect to direct form. Item forms that require this additional information processing step are, hypothetically, more difficult than those that do not. Another important distinction is that several of the problems contain numerical data unnecessary for the solution of the problem. Research has demonstrated (Hydle & Clapp, 1927) that problems containing superfluous data are more difficult than problems that do not. Finally, item forms can be distinguished on the basis of whether or not they require additional external rules for solution.

With these distinctions serving as guidelines, the item forms were divided into 24 groups which were then placed in a three-dimensional array. This 4 x 3 x 2 objective array is illustrated in Figure 1. Item forms in the top layer of the array (in cells 13-24) are those that require the application of one of the "indirect rules" while those in the bottom layer do not. Approximately the same number of item forms are in each layer. The three rows of the objective array represent the dimension that involves the number-of-direct-substitution data. Item forms in the first row require one or two direct substitutions, those in the second row require three, and those in the third row require four or five substitutions. These groupings were chosen so that almost the same number of item forms would lie in each row.

The last dimension of the objective array makes further distinctions between these six (2 layers by 3 rows) categories. Item forms in the first column require only direct substitutions in order to reach the solution. Those in the second column require the use of special multiplicative rules in addition to direct substitution. Item forms in the third column require the same processes as the second but also require the use of special external rules such as the temporal sequence equation. Finally, those item forms in the fourth column require the rules of the second column but also contain superfluous numerical data. The objective array, thus structured,

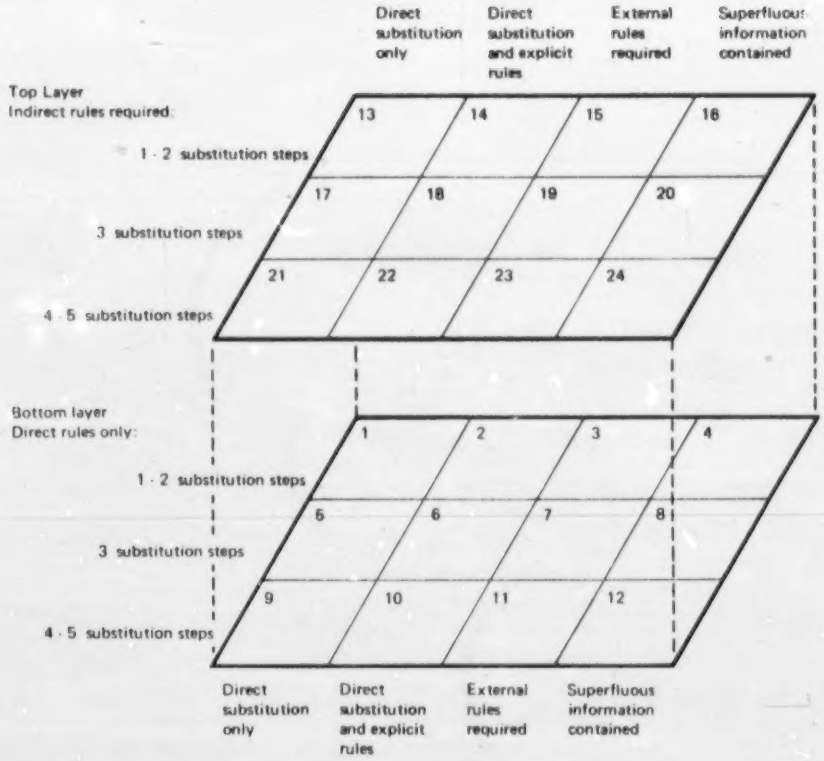

Figure 1. Array of instructional objectives for item form types.

groups all of the item forms in the program and organizes them according to prerequisite relationships.

Along each dimension of the array, the item forms increase in difficulty and build upon the preprequisite item forms that preceded. For example, in order to solve the item forms in the third column successfully, the student must possess the skills required to solve item forms in the first two columns. This is true for both layers and for every row. Further, to solve item forms in the second column, the student must possess the skills required to solve item forms in the first column. Similarly, this type of prerequisite relationship holds true for each dimension of the array. The exception is the fourth column. Only the first and second columns, but not the third column, are prerequisite to the fourth column.

Since the item forms within each group are nearly homogeneous in terms of the skills needed for solution, the groups of the array can serve as instructional objectives. For example, group 13 defines an objective as such: The student will be able to solve word problems that require one or two direct substitutions and application of an indirect rule. Graphically, the immediate prerequisites for a given objective are: (a) the objective in the layer below it, if any; (b) the objective to the left of it in the row, if any; and (c) the objective above it in the column, if any. A given objective also has as prerequisites all of the prerequisites of its prerequisites. For example, the immediate prerequisites of objective 18 are objectives 6, 17, and 14. In addition, objective 14 has as prerequisites 2 and 13; 17 has 5 and 13; 6 has 5 and 2; and 2 has 1 as a prerequisite. Thus, the prerequisites of objective 18 are objectives 1, 2, 5, 6, 13, 14, and 17. The objective array can therefore depict a quite complex learning hierarchy in a relatively simple fashion.

In practice, the instructional objectives of the Word Problem Program actually require the student to be able to solve a mix of problems from several groups. The rationale for this decision, which essentially seeks to

insure that students will not solve problems with trivial algorithms, will be discussed further in the next section. The method employed to accomplish this mix of problems required the designation of a "target" group and one or more "practice" groups. In the instructional session during the pilot test, the student received a set of problems randomly selected from the designated groups. Fifty to sixty-five percent of the problems were selected from the target group--the group for which the student's work is evaluated--and the remainder were selected from the practice group. For each group in the top layer of the array, the practice group was the group beneath the target group; for each group in the first row, the practice group was the group to the left; and, for each of the remaining groups, the practice group was the group immediately above it in the same column. In each case, the practice group is one of the previously mastered immediate prerequisites.

Students are evaluated on the basis of their performance on the problems from the target group. A student's work on a target group is designated as either mastery or nonmastery, and sequencing decisions are made on the basis of this evaluation.

The objective array is used to individualize the sequence of instructional objectives for each student. This sequencing is based on both the student's performance and on the hypothesized prerequisite relationships among the groups. After having mastered group 1, the student moves along a chosen dimension of the array until either the last group in that dimension is mastered or the student cannot master a given group. In either case, a new direction is chosen and instruction begins in like manner along that dimension. If progress has been stopped along all dimensions by nonmastery, instruction begins with the first nonmastered group as the target group. The program will first attempt to proceed across rows, then down columns, and finally, up to the top layer. This precedence is subject to change, however, depending on which direction is easiest for the student. For example, suppose a student mastered objectives 1 and 2 but failed on

objective 3. The program would then choose a new direction, and thus group 5 would be the next target group. If the student mastered objective 5 and proceeded to master objective 9, a new direction would again be chosen. If the student then failed objective 13, the nonmastered group 3 would be presented again. Failure here would again allow the student to proceed along the rows to objectives 6 and 10. Thus, although the student must eventually pass objective 3, initial or even subsequent failure on it will not impede progress in other areas of the array.

In summary, the solution algorithm and objective array specify instructional objectives, structure them in terms of their relationships, and are used to make sequencing decisions.

Instructional Strategies

All instruction in the Word Problem Program takes place by means of the computer. The program presents problems to the individual student at a time-sharing terminal, interacts with the student to assist in the solution process, judges the student's work, and chooses new problems appropriate to the student's state of learning. Students are expected to acquire the target problem solving skills by induction. Although specific instructional strategies are employed to assist in this acquisition, the sequence of instructional objectives bears the brunt of the instructional load. The Word Problem Program combines the traditional instructional methods that have shown the greatest success. This section outlines several instructional features of the program.

The program emphasizes translating the problem to number sentences. An analytic instructional method would encourage the student to proceed systematically with the translation. Several methods, which are all called analytic, have been studied. For example, Morton (1925) defines his analytic method as a three-step process, whereas Washburne and Osborne (1926) use a six-step process. All definitions, however, include three crucial

steps: (a) determine what is to be found; (b) determine what relationships and numerical data are given; and (c) decide which mathematical operations are needed to attain the solution.

In their informative review, Suydam and Riedesel (1969) conclude that "informal procedures are superior to following rigid steps . . . if the analysis method is used, it is recommended that only one or two of the steps be tried with any one problem" (p. 50). Rather than rely exclusively on analysis, a second method, relying on analogous problems, has been incorporated in the Word Problem Program.

The analogy method has a variety of definitions in the literature. All include two crucial parts: (a) a problem similar to the original is presented by the instructor; and (b) all the information necessary for the correct solution is preserved in the analogous problem (Gorman, 1967; Morton, 1925).

In the Word Problem Program, the analysis method and the analogy method are used when the student asks for a hint. The set of hints for each problem models the translation process.

The first hint for each problem identifies the unknown for the student and suggests that he or she reread the problem. This hint corresponds to the first step of the analytic method. The second hint is a restatement of the problem in simpler syntactic form, omitting superfluous information. The student is directed to compare the restatement to the original. This second hint corresponds to the analogy method. The third hint is a mathematical statement relating the variables in the problem and giving an appropriate number sentence. This hint corresponds to the second step of the analytic method by making the relationships between variables explicit. Since we assume students know how to solve number sentences, no hint corresponds to the third step of the analytic method. An example clarifies the relationship between hints and the original problem.

The Problem:

10 GROUPS OF WOMEN AND 19 GROUPS OF PUPILS WENT TO THE BALL PARK. HOW MANY MORE PUPILS THAN WOMEN WERE THERE AT THE BALL PARK IF THERE WERE 9 PEOPLE IN EACH GROUP?

First Hint:

HINT: RESTATE THE QUESTION. FIND THE NUMBER OF MORE PUPILS THAN WOMEN AT THE BALL PARK.
NOW REREAD THE PROBLEM.

Second Hint:

HINT: RESTATE THE PROBLEM. IT IS SIMILAR TO:
10 GROUPS OF 9 WOMEN AND 19 GROUPS OF 9 PUPILS WERE AT THE BALL PARK.
? = MORE PUPILS THAN WOMEN.
NOW REREAD THE ORIGINAL PROBLEM.

Third Hint:

HINT: TRANSLATE THE PROBLEM TO A NUMBER SENTENCE:
10 x 9 WOMEN + ? WOMEN = 19 x 9 PUPILS, OR,
10 x 9 + ? = 19 x 9
NOW, SOLVE THE NUMBER SENTENCE.

The hint structure used in the program is designed to encourage the student to take the following steps in problem analysis: (a) identify the unknown; (b) translate the word problem into a simpler analogous problem; (c) translate the simpler problem into an arithmetic number sentence; and (d) solve the number sentence. This structure combines the best features of the analytic and analogy methods, providng help when it is needed and modeling desirable strategies for solving word problems.

One final contingency must be provided for in order to ensure the success of the program. This contingency deals with the subtle distinctions between problem solving and algorithmic behavior.

In most elementary school curricula, word problems are used as exercises to increase competence in computation. For example, a lesson designed to teach the student to multiply three-digit numbers will often be followed by a set of word problems that requires the multiplication of three-digit numbers; the student can solve the problem without reading a single word. In their report on research and development in elementary school mathematics, Suydam and Riedesel (1969) state that studies "reveal that pupils often give little attention to the actual problems; instead they almost randomly manipulate numbers" (p. 47). R. P. Stevenson (cited in Riedesel, 1967) pointed out that manipulation of the numbers is more algorithmic than random. For example, he describes a method used by an elementary school student:

> If there are lots of numbers, I add. If there are only two numbers with lots of parts, I subtract. But if there are just two numbers with one littler than the other, it is hard. I divide if they come out even, but if they don't, I multiply. (p. 308)

In extreme cases, the numbers alone cue the correct operation, and the student responds algorithmically rather than by utilizing problem solving skills. The order in which problems are presented can also radically alter the difficulty of individual problems. For example, both Loftus (1970) and Hydle and Clapp (1927) found problems are easier when they can be solved by the same operations (in the same order) as the preceding problem. Students often respond to a new problem by attempting to apply the solution steps from the previous problem.

To avoid reinforcing such undesirable algorithms, the Word Problem Program employs three strategies. First, the problems presented by the program are written so that no word or phrase consistently cues the correct operation. For example, the words "divided by," "times as many as,"

"each," and "average" are used in both multiplication and division problems. Second, problems are sequenced so that students never see a set of problems that can all be solved with the same operation. This discourages responding with the operation used in the preceding problem. Finally, the numbers in the Word Problem Program are chosen so that they vary in magnitude so that one of the numbers is a factor of another, regardless of which arithmetic operation correctly solves the problem. This feature eliminates the cues that the numbers themselves provide and, along with the first two features, forces the student to find more appropriate strategies for choosing a particular operation.

A goal of the Word Problem Program is to teach the translation of a word problem to a number sentence, but not to teach the solution of the equations or the computation of the final answer. For this reason, as well as to allow the freedom to choose numbers by the above criteria, the program must do the arithmetic calculation for the student. The Word Problem Program requires that the student specify the operations and operands necessary to solve the problem, but does not assume that they have the ability to compute with the large numbers used in the problems.

The Pilot Version

The Word Problem Program operated on a tryout basis at the small computer resource at an elementary school in suburban Pittsburgh. The computer (a DEC PDP 15 with 64K 18-bit words of memory) has been located in a van outside the school since Spring 1972, and now provides locally controlled service to the school throughout the day. In spite of the computer's small size, it has a general-purpose time-sharing system (ETSS) designed and implemented by the Learning Research and Development Center (Fitzhugh, 1970). ETSS currently supports 16 terminals.

A student's interaction with the Word Problem Program begins with a problem written on the screen of a cathode-ray tube. The numeric data

presented in the problem is then listed beneath the problem (in order of appearance in the problem) and labeled with consecutive alphabetic characters, each on a separate line. After reading the problem, the student has several options available: He or she may type "CHANGE" to get a new problem, type "HINT" to receive a hint, type "ANSWER" followed by a letter to indicate that the number represented by the letter is his/her answer, or perform an arithmetic operation. To perform an operation, the student types two operands (represented by alphabetic symbols) separated by an operation symbol (+, -, x, or /). The computer then performs the calculation and displays the result labeled with the next available alphabetic character. Each time the computer processes a request, it erases the request and any superfluous displays from the screen before it creates the new display.

If the student types an incorrectly formatted request, the program diagnoses the mistake and provides corrective feedback in the form of messages. For example, if the student typed "19 + 24" the program would remind him that he should "TYPE LETTERS ONLY". If the student designates an incorrect value for the answer, the program types "WRONG ANSWER" followed by the prompt "TYPE 'HINT' IF YOU WANT SOME HELP"; if the answer is correct, a bell is sounded and the message "GREAT!" is typed on the screen.

Pilot Test

Throughout the developmental period, the Word Problem Program has undergone formative evaluation. The program has been reviewed by other members of the Computer-Assisted Instruction in Problem Solving Project in each of its design and experimental phases. It has also been tested on four students in an experimental non-computer version and with four students in an experimental computer version that required intervention by the experimenter. Both of these tryouts were conducted in the LRDC laboratories.

The results of these evaluations helped refine the program and encouraged continued developmental efforts. This section reports the results of a third pilot test which was conducted in a school setting with minimal intervention by the experimenter.

The pilot test was performed with two fourth-grade and three fifth-grade students at an elementary school. Four of the students were chosen by their teachers for participation in the pilot test on the basis of their need for remedial instruction in word problems. The fifth student was chosen because his teacher felt he lacked motivation, required special attention, and was disruptive in the normal individualized classroom environment.

Each student received instruction from the program for two sessions per week over a seven-week period. The sessions lasted from 10 to 40 minutes, with a norm of approximately 25 minutes. The first session consisted of an introduction to the format of the program and a 10-item pretest which sampled ten cells from the array. In the last session, the pretest was repeated using the same item forms.

The sessions were held during the time set aside each day for the students to engage in activities of their choice, provided they have fulfilled their weekly commitments. Students left their rooms at the specified time to come to the centrally located terminal area for instruction. Students had their sessions individually with the experimenter sitting beside them at the terminal.

The experimenter assessed the student's performance and chose the next instructional objective, since these features had not yet been programmed. Interactions with the students were limited to casual conversation, instructions as to which objective to ask for (this occurred once or twice a session), and, occasionally, questions concerning the student's rationale for any particularly interesting behavior. Anecdotal records of the student's solicited and unsolicited verbalizations were taken by the experimenter each time they occurred. These records supplemented the

complete history of every interaction between the student and the program, which was taken by the program.

In this stage of formative evaluation, we sought answers to the following questions:

(a) Could students learn to respond quickly and efficiently in the format required by the program?

(b) Could students solve a greater variety of problems after the instructional period than before?

(c) Did the hypothesized hierarchical relationship hold true for the students studied?

(d) Did students enjoy and value the experience of working on the program?

Obviously, this list does not exhaust the questions we intend to answer concerning the program. Because of the time and programming limitations, however, we were forced to delay the evaluation of the extent to which the program accomplishes its problem solving goals, the effectiveness of each component instructional strategy and program feature, the empirical validity of the array, and many more. The remainder of this section presents data that provides tentative answers to the four questions stated above.

Students were able to master a set of number sentences and simple introductory problems in 10 to 15 minutes during the first session. These problems were heavily supplemented by instruction from the experimenter. After this introduction, no additional instruction was given in the response format. The students made few errors in inputting commands during subsequent sessions. Thus, students were able to learn the response format quickly, easily, and effectively. The ease with which students mastered the command format contrasts with the earlier experience of Loftus (1970)

with a computer-based word problem test. In her work, students took four to eight weeks to master the response format. The Loftus work is described in the design document for the Word Problem Program (Roman & Laudato, 1974), and the changes we made to improve the response format are detailed there.

To answer the second and third questions posed above, we can make use of the pre- and posttest data collected by the program. Figure 2 graphically depicts the ten cells of the array which were sampled for the tests.

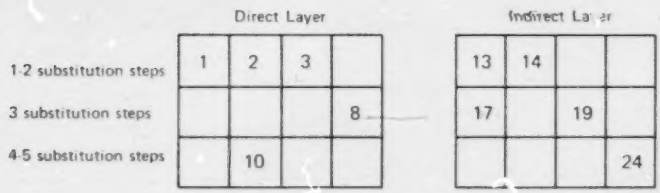

Figure 2. Cells of the array sampled in the pretest.

The results of the pretest, instructional sessions, and posttest are summarized in Figure 3 for all five students. Each rectangle represents a layer of the array; the direct layer is pictured below the indirect layer. A solid black box within a rectangle represents mastery of the problem or objective; an "X" represents nonmastery; and a blank means that the problem or objective was not tried. This latter case occurred only once. Student One's performance on the first six problems in the pretest was so poor that, to avoid alienating him, the session was terminated.

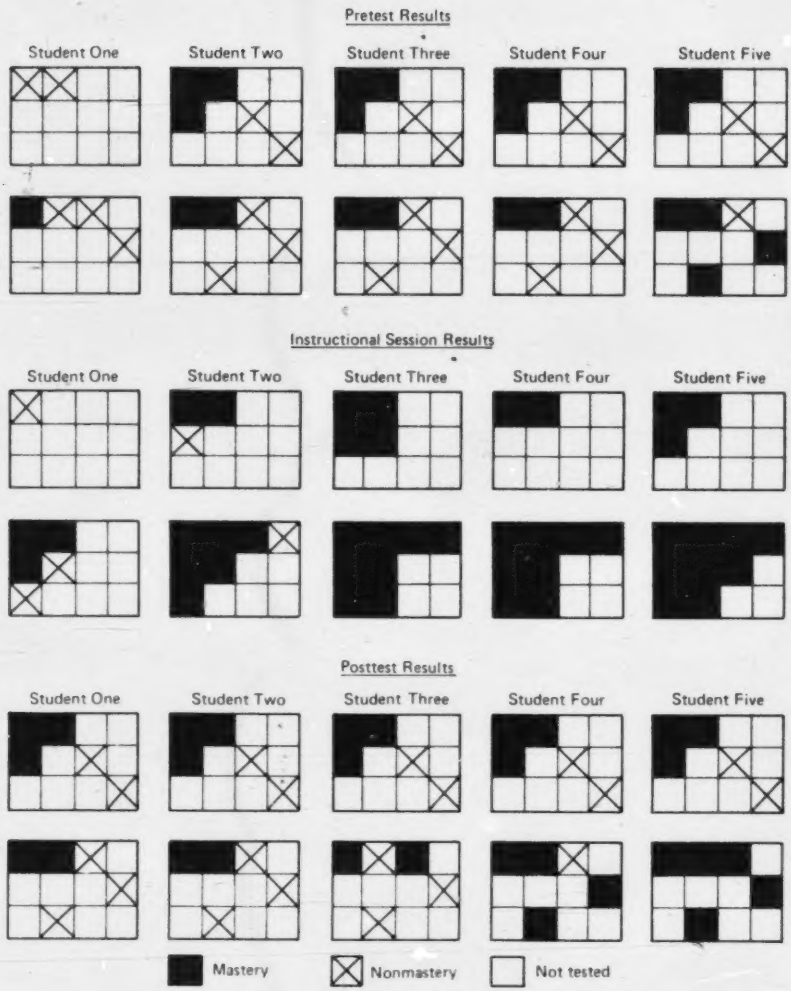

Figure 3. Tryout data.

The set of rectangles labeled "Instruction" depicts the status of the array for each student at the end of the instructional period. An "X" in a box of one of these arrays indicates that the student worked one or more times on that objective but failed to meet the mastery criterion. It should also be noted that a single solid black box may represent several instructional sessions during which the student attempted to master the objective. The graphs depict a positive change in the students' problem solving behavior as measured along the dimensions of the array. While such a small sample cannot definitively prove the program works, these results are encouraging.

In all cases but that of Student Two, the posttest performance was superior to the pretest performance. Student Three made an error on a problem (from cell 2) on the posttest that she answered correctly on the pretest. This is even more unusual given that, from her instructional array, she had mastered objectives which have, as prerequisites, all of the skills needed to solve that particular problem. Of the 11 problems from cell 2 that Student Three saw during instruction, 10 were performed errorlessly. The student clearly should have been able to solve the problem on the posttest. There were three other disparities between the posttest data and the data from the instructional sessions. In these cases, the student demonstrated mastery of a particular cell, but answered the posttest problem from that cell incorrectly.

A dramatic increase in performance was demonstrated by Student One, a fifth-grade student who experienced difficulty on the pretest and in the first few instructional sets. By the end of the instructional period, the student's performance was equal to the entry level of the other students.

A rearrangement of the pre- and posttest data will help in answering the third question concerning the hierarchical relationships between cells of the array. Figure 4 depicts the results on both tests for all students in ten 2 x 5 rectangles--one for each problem from the test. Each

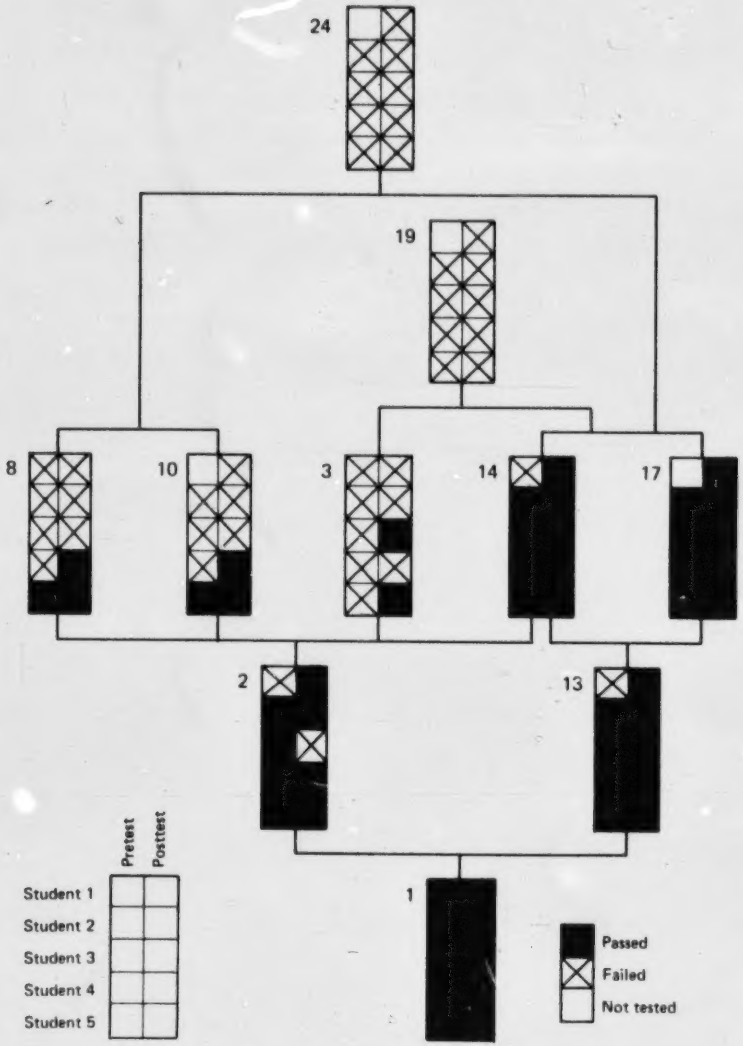

Figure 4. Prerequisite relationships between problems.

rectangle also contains two columns: The first represents the pretest score and the second the posttest score. As above, a solid black box represents a correct answer on the problem, an "X" represents an incorrect answer, and a blank indicates that the student did not receive that problem on the test.

The rectangles (problems) are connected by lines to illustrate the relationships between problems. One can consider each problem as representing the instructional objective as such: The student should be able to solve problems requiring the information processing skills stipulated by the problem's position in the array. The figure is constructed using the hierarchical relationships between objectives in the array. Thus, problems that lie below a given problem are prerequisite to it. They are not always the only prerequisites, however, and thus the figure does not represent a conventional learning hierarchy.

For the given set of data, the hypothesized prerequisite relationships are validated if the following condition holds true: If a particular problem has been mastered, then all problems beneath it in the figure should have been mastered, or equivalently, if a given problem was not mastered, then no problems above it in the figure should have been mastered. The rectangles contain data for ten different cases, that is, a pre- and a posttest for each of the five students. Thus, Figure 4 represents ten different tests of the validity of the relationships between problems. Of the ten cases, nine perfectly meet the condition stipulated above. The exception is Student Three's posttest performance on problem 2, which was discussed previously. Even if this exception could not be rationalized, the results would still be convincingly positive.

In order to answer the fourth question, data was collected informally throughout the tryout period. The students' attitudes and motivations were noted during each session, and each comment the students made with respect to the program was recorded. The attitudinal and behavioral

data will be reported first, followed by discussion of the comments of each individual student.

Of the 62 sessions for the five students, there were seven (11%) during which the student behavior fell to simply being willing to receive; during 45 sessions (73%), the students were willing to respond; and in the remaining ten sessions (16%), the students exhibited satisfaction in response. The next paragraphs make the behaviors exhibited more precise.

In sessions that were judged as willing to receive, the students were either inattentive, not task oriented, or behaved indifferently towards the program. When called upon to come to the session, they came reluctantly or slowly.

In sessions judged as willing to respond, the students came early for their work, waited impatiently for the previous student to finish, or left their previous task quickly. Students attended to the display carefully. Conversation dealing with topics other than the program was minimal and usually limited to the beginning and end of the session. The students generally smiled and appeared relaxed and happy with the work.

In the sessions judged as satisfaction in responding, the students were even more positive. They hurriedly and excitedly came to the session. They smiled, laughed, and joked a good deal during the session, but were attentive to the task. They often read the problems aloud with enthusiasm and laughed at the occasional rhyming problems or those with many or large numbers. They were obviously enjoying themselves greatly. Most protested when told the session was over.

The anecdotal data was also largely positive. This data will be examined here student by student. Student One's reactions were mixed. During the introductory set of problems he exclaimed, "This is neat!" On the second day, he came running out of his classroom to the terminal center and shouted, "I've been waiting for this day!"

Between the second and third session, Student One worked on the program by himself, apparently in his free time. This behavior was discouraged, but indicates his interest and enthusiasm. On later occasions, he complained about both indirect problems and multi-step problems: "I don't like the ones like that" and "I don't like these problems." During the ninth session, he refused to work on any multi-step problems--"just for today." He also complained that his work on the Word Problem Program interfered with his required school work--"I'm not getting my work done"--and even missed one session in an attempt to make up classroom work, after a one-week absence due to illness. However, he also complained in session three when the experimenter missed a day, and in session four, he pleaded for an extra session to make up for the one the experimenter missed. He suggested we have the make-up session at lunch time or after school. Twice he indicated that he preferred to work on the introductory set and the number sentence problems: "They're O. K. I liked the other ones better" and "Now can we do number sentences?" Finally, in one instance, he refused to terminate the session by typing "STOP". Instead, he quickly pressed RETURN to get a new problem before the experimenter could intervene. Although Student One was the only student to offer negative comments, he made the greatest gain in posttest performance.

Student Two's comments cannot be taken literally. The smile on her face and the laugh in her voice indicate her true feelings more than the words she used. In one instance, she laughingly stated, "This computer's wacky" and repeated her message to several passing students. She indicated her pleasure in solving problems with large numbers on several occasions and twice demanded, "I want big numbers." She also jokingly responded to a series of several problems which contained many numbers with, "Ooh, I hate these kind."

Student Three, although she read virtually every problem aloud (perfectly and rapidly), did not offer many spontaneous comments. One anecdote reveals her strong feelings towards the program. When the experimenter arrived at her homeroom for the fourth session, he found a party in progress. Class had been cancelled and the students were playing games and eating candy. When she saw the experimenter, she immediately prepared to leave for the session. Her teacher interrupted and told her that she did not have to go on the computer today; the experimenter agreed, indicating that they could make it up another day. She insisted, however, that she wanted to have the session and would return to the party afterwards.

Student Four seemed to enjoy the program more than anyone else. On several occasions, she verbally indicated her enjoyment of both rhyme problems and multi-step problems. On two occasions, she smiled and commented--"Oh, I hate these kind"--in reference to more difficult problems. Once she objected when told the session was over and prolonged it an extra 15 minutes to almost 40 minutes--the longest session in the experiment.

Student Five also was positive about the program. Like Student One, he insisted that we make up for the lost day at lunch time. One conversation in particular, however, illustrates his feelings toward the program. Between problems in the fourth session, he asked the experimenter, "Did you like school when you were a little kid?" The experimenter responded, "Sometimes," and Student Five exclaimed, "Me, never--but I like this. I really like this!"

In summary, all students seemed to respond to the program at a level which is at least equivalent to Krathwohl's "Acquiesence in Responding" (Krathwohl, Bloom & Masia, 1964): The student is active in responding but passive in the initiations of behavior. There are many instances of "Willingness to Respond": The student chooses to engage in activities

within the program and in problem solving. There are some instances of "Satisfaction in Response": The student demonstrates "a feeling of satisfaction, an emotional response, generally of pleasure, zest, or enjoyment" (Krathwohl et al., 1964, p. 130).

Conclusions

The results of the pilot test provided useful feedback for the redesign and refinement of several program components. Positive results were attained for each of the four questions asked at this stage of formative evaluation. A more detailed study is currently underway which seeks to answer other questions in a more rigorous fashion. This study involves 28 students working on a version of the program requiring no intervention by the experimenters.

By far, the most intriguing product of our work on the program is the algorithm and array. The solution algorithm represents a methodology for performing a rigorous component analysis of complex cognitive tasks. This allows the precise specification of learning objectives and the prerequisite relations between them. The objective array represents a methodology for organizing the data from a component analysis and for making sequencing decisions which facilitate a high degree of individualization. We believe that these methodologies are generalizable to other situations involving complex cognitive tasks. We are beginning to investigate such an application to the design of a curriculum to teach competency in solving series. This investigation and other studies should provide data to assess the true significance of the methodologies.

References

Block, K. K., Carlson, M., Fitzhugh, R. J., Hsu, T., Jacobson, E., Puente, G., Roman, R. A., Rosner, J., Simon, D. P., Glaser, R., & Cooley, W. W. A computer resource for the elementary school: Progress report 1971-1972. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1973. (LRDC Publication 1973/1; ERIC Reproduction Service No. ED 076 304)

Bobrow, D. G. Natural language input for a computer problem-solving system. In M. Minsky (Ed.), Semantic information processing. Cambridge: Massachusetts Institute of Technology, 1968.

Fitzhugh, R. J. LRDC experimental time-sharing system reference manual (3 vols.). Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1970.

Gorman, C. J. A critical analysis of research on written problems in elementary school mathematics. Unpublished doctoral dissertation, University of Pittsburgh, 1967.

Hively, W., Jr. Defining criterion behavior for programmed instruction in elementary mathematics. Unpublished doctoral dissertation, Harvard University, 1963.

Hydle, L. L., & Clapp, F. L. Elements of difficulty in the interpretation of concrete problems in arithmetic. Madison: University of Wisconsin, Bureau of Educational Research, 1927. (Research Bulletin No. 9)

Krathwohl, D. R., Bloom, B. S., & Masia, B. B. Taxonomy of educational objectives, handbook II: Affective domain. New York: David McKay Company, 1964.

Laudato, N. C. The design and assessment of a computer-assisted instructional program in word problem solving for the elementary school. Unpublished doctoral dissertation, University of Pittsburgh, 1975.

Loftus, E. J. F. An analysis of the structural variables that determine problem solving difficulty on a computer-based teletype. Stanford: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1970. (Technical Report No. 162)

Morton, R. L. An analysis of errors in the solution of arithmetic problems. Educational Research Bulletin, 1925, 4(9), 187-190.

Newell, A., & Simon, H. A. Human problem solving. Englewood Cliffs, N. J.: Prentice-Hall, 1972.

Olton, R. M. A self-instructional program for developing productive thinking skills in fifth and sixth-grade children. Journal of Creative Behavior, 1969, 3, 16-25.

Osburn, H. G. Item sampling for achievement testing. Educational and Psychological Measurement, 1968, 28, 95-104.

Paige, J. M., & Simon, H. A. Cognitive processes in solving algebra word problems. In B. Kleinmintz (Ed.), Problem solving. New York: Wiley, 1966.

Papert, S. Teaching children thinking. In Mathematics teaching. Leicester, England: The Association of Teachers of Mathematics, 1972.

Polya, G. How to solve it. Garden City, N. Y.: Doubleday, 1957.

Research for Better Schools, Inc., & Learning Research and Development Center. IPI mathematics continuum chart. New York: New Century Education Corporation, 1972.

Riedesel, C. A. Guiding discovery in elementary school mathematics. New York: Appleton-Century-Crofts, 1967.

Roman, R. A., & Laudato, N. C. Computer assisted instruction in word problems: Rationale and design. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1974. (LRDC Publication 1974/19)

Suydam, M., & Riedesel, C. A. Interpretative study of research and development in elementary school mathematics (3 vols.). Final Report. College Park: Pennsylvania State University, 1969.

Washburne, C., & Osborne, R. Solving arithmetic problems. II. The Elementary School Journal, 1926, 27, 296-304.

Wickelgren, W. How to solve problems: Elements of a theory of problems and problem solving. San Francisco: W. H. Freeman, 1974.